Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m,k)-firm model *

Jian Li
LORIA-INPL
Jian.Li@loria.fr

YeQiong Song
LORIA-UHP Nancy 1
Song@loria.fr

Françoise Simonot-Lion
LORIA-INPL
Simonot@loria.fr

LORIA, Campus Scientifique - BP 239, 54506 VANDOEUVRE-LES-NANCY, FRANCE

Abstract

(m,k)-firm model has recently drawn a lot of attention. It provides a flexible real-time system with graceful degradation of the QoS, thus achieving the fault tolerance in case of system overload. In this paper we first give a review of the existing work on the use of the (m,k)-firm model for handling the QoS and fault tolerance management. Then we focus on DBP algorithm as it presents the interesting feature of dynamically assigning the priorities according to the system's current state (QoS-aware scheduling). However, DBP cannot readily be used for systems requiring deterministic (m,k)-firm guarantee since the schedulability analysis was not done in the original proposition. In this paper a sufficient schedulability condition is given to deterministically guarantee a set of periodic or sporadic activities (jobs) sharing a common non-preemptive server. This condition is applied to two case studies showing its practical usefulness for both bandwidth dimensioning of the communication system providing graceful degradation of QoS and the task scheduling in an in-vehicle embedded system allowing fault tolerance.

1. Introduction

It is well known that real-time systems designed according to the worst-case condition (case of hard-real time system design) often result in large resource requirement. As at run time the system is seldom in a worst-case condition, a large amount of system resources are under-utilized. One solution is to design the system based on an average case. This solution can be suitable for a subclass of soft real-time systems requiring only probabilistic deadline guarantee. However, for other real-time systems such as those found in multimedia and the automatic control domain, providing only probabilistic deadline guarantee can be unacceptable. A more precise specification on how the deadline misses are distributed in time is necessary [2] and this can be done using the (m,k)-firm model [4]. Typically, for the same deadline miss ratio, a real-time application better tolerates non-consecutive deadline misses than consecutive ones. A system is said under (m,k)-firm real-time constraint if it requires the guarantee of the deadline meet of at least *m* out of any *k* consecutive invocations of a recurrent job.

Much previous work has dealt with new scheduling algorithms integrating the additional (m,k)-firm constraint [10]. Two families can be found: *dynamic* and *static*. DBP (Distance Based Priority) [4] and DWCS (Dynamic Window-Constrained Scheduling) [12] are dynamic. The priority assignment done on-line is based on the current state of the system. ERM (Enhanced Rate Monotonic) [9] and EFP (Enhanced Fixed-Priority) [8] are static as the scheduling is done off-line using a static deadline miss pattern. These four algorithms will be briefly discussed in section 2.

Finally, note that, as for hard real-time, sufficient conditions of feasibility are obviously required in order to ensure a *deterministic (m,k)-firm guarantee*. There are sufficient conditions for ERM, EFP [9] and DWCS [12] [13], but no such condition has been investigated for DBP.

The rationale for only considering the dynamic (m,k)-firm scheduling algorithms in our work, is based on the following reasons. The system should be able to adapt to workload variation (e.g. in networks handling QoS with connection admission control) by taking advantage of the possibility to *discard* until *k-m* out of *k* consecutive jobs during system overload periods. So, in this context, off-line scheduling is simply not suitable. Furthermore, the use of a dynamic scheduling policy rather than a static one, allows a better exploitation of the available resources in general. Finally we insist on the importance of discarding the instances of jobs whose deadlines cannot be met by the system. In fact, an overload situation leads to deadline misses, and only discarding part of jobs (preferably those with missed deadlines) allows better managing it. Scheduling with dynamic job drops makes our work

* This work is supported in part by the REMPLI EU project (www.rempli.org). It was presented in part at the 5th IEEE international workshop on factory communication systems (WFCS2004), Vienna (Austria), Sept. 22-24, 2004

different to the classic scheduling studies without drops (e.g. [2], [3], [16]).

Once we have established that a dynamic policy is better suited to the application requirements, we have to justify the choice of DBP in our work as opposed to DWCS. We recall, that for the targeted applications, we have to exhibit at least a sufficient feasibility condition. For DWCS such a condition was established in [12]; but it has a limited application region since the jobs must be with the same service time and the same periods. That is why, although DBP itself could be improved [7], we investigated this scheduling in order to find a more general condition. Moreover as we would like to obtain a result applicable to both CPU task scheduling and network packet scheduling, we further restrict ourselves to *non-preemptive scheduling*. As proposed in former studies [4], [12], we consider EDF for equal priority cases.

Therefore in this paper we only focus on NP-DBP-EDF (Non-Preemptive - Distance Based Priority – Earliest Deadline First). The rest of the paper is organized as follows: Section 2 describes the problem and outlines the related work; Section 3 presents the sufficient schedulability condition under NP-DBP-EDF, which is the main contribution of this paper. In Section 4, we demonstrate how this condition can assist the designer for efficiently dimensioning a system. The results obtained in two case studies are compared with those obtained, within the limits of (k,k)-firm (or equivalently hard real-time), from the sufficient condition presented in [5]. The limits of the deterministic (m,k)-firm guarantee are also discussed, highlighting the need of another real-time constraint model. Finally we conclude our findings in Section 5.

2. Problem description and related work

2.1 System model

Consider the following MIQSS (Multiple Input Queues Single Server) model (Fig. 1) where a set of n jobs (or streams) $a() = \{a(1),a(2),...,a(n)\}$ share a single server of capacity c.

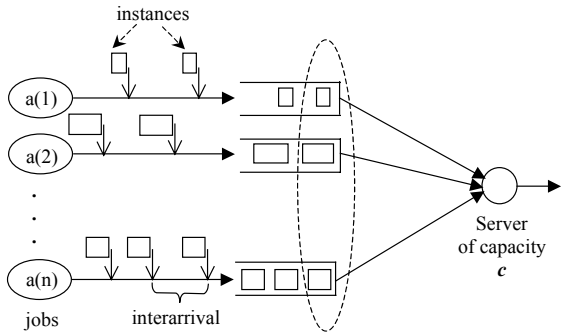

Fig. 1: MIQSS model

We consider the jobs that are periodic or sporadic with p_i as the period or minimum inter-arrival time and a deadline $d_i = p_i$. An instance (or invocation) of *a(i)* requires an execution time of c_i. Each job *a(i)* is assumed to be under a (m_i,k_i)-firm constraint ($m_i < k_i$). So, a job *a(i)* can be modelled by (c_i, p_i, m_i, k_i).

A job under (m, k)-firm constraint can be found in one of the two following states: *normal* and *dynamic failure* [4]. To find out the current state of a job we need to examine the execution history of the last k instances. If we associate '*1*' to an instance with a deadline met and '*0*' to an instance with a deadline miss, this history is then entirely described by a group of k bits called *k-sequence*. The system fails into a dynamic failure state when any job's k-sequence contains less than m '1'. Fig. 2 shows the state transition diagram for (2,3)-firm; the left-most bit in the group represents the oldest event. Each new instance arrival causes a shift to the left in the group, the left-most exits from the k-sequence and is no longer considered, while the right-most will be a *1* if the instance has met its deadline or a *0* if not.

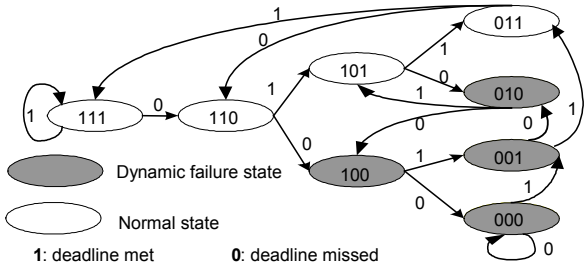

Fig. 2: State-transition diagram with (2,3)-firm

2.2 Distance Based Priority (DBP) strategy

DBP together with the concept of (m,k)-firm, was first introduced by Hamdaoui and Ramanathan [4] for scheduling a set of job streams sharing a common server. DBP dynamically assigns priority to the jobs of a stream according to the *distance* of the current k-sequence to a dynamic failure state. The closer the stream is to a failure state the higher its priority is. This distance can be easily evaluated, by adding *0*s to the right side until the failure state and the number of added *0*s is equal to the priority. The k-sequence can be considered, in a way, as a kind of on-line QoS measurement system and thus DBP can be seen as a dynamic scheduling policy with feedback. The resulting priority then contributes to maintaining the global performance of the system. Results obtained from simulation [4], [7], [11] have shown that DBP provides good statistic performance, which can be used for applications requiring statistic (m,k)-firm guarantee. However, for applications requiring deterministic (m,k)-firm guarantee, we need a sufficient schedulability condition.

2.3 Related work

In [9], an off-line fixed-priority algorithm called ERM is proposed and the corresponding sufficient schedulability condition is given. Instances of a job are first classified as mandatory (1) and optional (0), providing a fixed k-sequence to indicate its (m,k)-firm constraint. Nevertheless, satisfying an (m,k)-firm constraint using the fixed k-sequence is more restrictive than actually needed and could potentially result in more resource requirement. Moreover, in the MIQSS model, several k_i-sequences ($i = 1, ..., n$) could concentrate their mandatory instances on the time axis, resulting in a peak workload for the server. In [8], the Worst Case Interference Point (WCIP) is defined for a job of priority i. It is the time instant at which the maximum execution interference from higher priority job set may occur. Then EFP is proposed to reduce WCIP. It consists in rotating the k-sequences (or (m,k)-patterns) according to a heuristic approach. It has been shown that finding the optimal superposition of k_i-sequences is NP-hard in strong sense. It is also true for any dynamic algorithm. Thus, neither DBP nor DWCS can be optimal.

Contrarily to DBP which only uses the distance, DWCS [12] dynamically assigns priority to job *a(i)* based on $W_i = x_i/y_i$. It ensures that in every fixed window of y_i consecutive instances, a minimum of y_i-x_i instances must meet their deadline. Otherwise a service violation occurs (dynamic failure). In DWCS, instances will be either executed before their deadlines or dropped as in DBP. Furthermore, even though DBP works in a sliding window while DWCS does in a fixed window, they have the equivalence since they can be transformed to each other.

Mok and Wang [15] have proven that in general, DWCS can fail for arbitrarily low workload. The sufficient schedulability condition, given in [12] for DWCS, has improved the utilization factor, but jobs must have the same periods and execution time.

Intuitively, DBP [4] constitutes a more efficient solution and it potentially requires less server capacity than the static algorithms. But the schedulability analysis of a dynamic algorithm is difficult. In [5], a necessary and sufficient schedulability condition for HRT is given for a set of periodic or sporadic jobs with arbitrary release time. A job *a(i)* is modelled by (c_i, p_i) with $d_i = p_i$. Time is assumed discrete and clock ticks are indexed by natural numbers. Job invocations and executions only start at the clock ticks; each of the parameters c_i and p_i are expressed as multiples of clock ticks.

Jeffay's theorem: Let $a() = \{a(1), a(2), ..., a(n)\}$ be a set of *sporadic or periodic* jobs sorted in non-decreasing order by periods (*i.e.*, for any pair of jobs *a(i)* and *a(j)*, if $i > j$, then $p_i \leq p_j$). If *a()* is schedulable then

$$(1) \quad \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$$

$$(2) \quad \forall i,\ 1 < i \leq n;\ \forall L,\ p_1 < L < p_i:$$

$$c_i + \sum_{j=1}^{i-1} \left\lfloor \frac{L-1}{p_j} \right\rfloor c_j \leq L$$

And if *a()* satisfies conditions (1) and (2) then the non-preemptive EDF scheduling algorithm will schedule any concrete set of periodic or sporadic jobs generated from *a()*.

End of theorem

This result could be used to give a more restrictive sufficient condition for (m,k)-firm constraint. In fact, when m is equal to k, an (m,k)-firm constraint becomes hard real-time. In this case, DBP does not work and only EDF is used. However, the server capacity dimensioned using this condition might be oversized, as it does not drop k-m out of any k consecutive instances. So, in order to deal with this problem, we apply a rationale similar to that done in Jeffay's theorem to obtain a sufficient condition for NP-DBP-EDF. This is the purpose of the next section.

3. Sufficient condition for NP-DBP-EDF

Unlike hard real-time (HRT) scheduling, with (m,k)-firm scheduling there is not a condition which is both necessary and sufficient. In [7] we have given a necessary condition for NP-DBP-EDF. But this necessary condition only tells us that meeting all (m_i, k_i)-firm constraints is impossible if the server capacity is below a certain threshold. It does not tell us what the sufficient server capacity is for meeting all (m_i, k_i)-firm constraints. Therefore, for providing deterministic guarantee, a sufficient condition is fundamental.

3.1 NP-DBP-EDF scheduling algorithm

DBP is used to decide which one of head-of-queue job instances should be served at first in the MIQSS model. In case of the same DBP value, EDF is used. We note by $DBP_j(t)$ the DBP value of job j at time point t. Under the NP-DBP-EDF scheduling policy, at time t, the instance, which is being executed in the unique server, has highest priority because of the non pre-emption. Instances of a same job are stored in a FIFO queue. The instances waiting for execution at the head of the queues at time t are served in the order of their DBP priorities. The instances with $DBP_j(t) = 1$ (*for* $j = 1, 2, n$) must be executed before their deadlines, otherwise their (m_j,k_j)-firm constraint guarantee will be violated. Instances with $DBP_j(t) > 1$ will be executed if they can have the server and the operation be completed before the deadline, otherwise they are discarded. The fact of discarding job instances makes the following schedulability analysis different to the classic ones (e.g. those found in [2], [16]).

3.2 Busy period and workload evaluation

We define **DBP=X busy period** as the time interval during which the server is occupied by, and only by, the instances of jobs whose DBP value is equal to **X**.

Obviously, any missed deadline of DBP=1 instance will violate the (m,k)-firm constraint. This is the reason why, afterwards, we will only focus on the worst-case processor demand relative to DBP=1 busy period.

According to NP-DBP-EDF scheduling, except for the running instance (non pre-emption), DBP=1 instances have the highest priority. So, once there are DBP=1 instances, they should be executed immediately or simply wait until the end of the executing instance.

First, we consider the situation that, at one time point, DBP=1 instance appears and can have the server immediately. The workload is calculated in the following DBP=1 busy period, giving the worst-case possibility. Let t_0 be the starting time of this DBP=1 busy period in this situation, t_d the ending time, and let $L = t_d - t_0$ be the length of the DBP=1 busy period.

Jobs are divided into two sets: one is for the jobs whose DBP=1 instances start from the beginning time of DBP=1 busy period, denoted by U. The other set is $a()$-U.

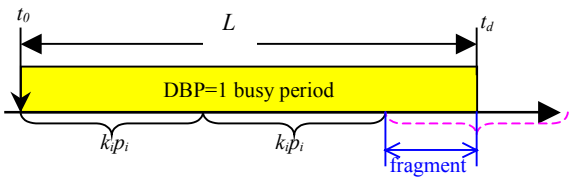

Fig. 3: Workload of DBP=1 instances starting at time point t0.

As shown in Fig 3, it is given that in every interval k_ip_i for the job $a(i) \in U$, there are m_i and only m_i instances of job $a(i)$ with DBP=1. Only these instances can be executed and meet their deadlines. This generates a workload of:

$$W_U^1 = \sum_{i \in U} \left(\left\lfloor \frac{L}{k_i p_i} \right\rfloor m_i \right) c_i \qquad (1)$$

But, in general, interval L is not an integer multiple of k_ip_i. So, in the *fragment* (the residue of L divided by k_ip_i), for a job $a(i)$, the number of possible instances is bounded by m_i. This results in the following term:

$$W_U^2 = \sum_{i \in U} Min\left(\left\lceil \frac{L - k_i p_i \left\lfloor \frac{L}{k_i p_i} \right\rfloor}{p_i} \right\rceil, m_i \right) c_i \qquad (2)$$

By using equations (1) and (2), the workload caused by all jobs in U is then:

$$W_U = W_U^1 + W_U^2 \qquad (3)$$

The workload caused by the second part ($a(j) \in a()$-U) is calculated as follows. Jobs not included in set U have their DBP value greater than 1 at the time point t_0. It is clear that in DBP=1 busy period, only DBP=1 instances can be executed. So, Fig. 4 shows how a job $a(j)$ starts to generate the workload from t_0 in DBP=1 busy period.

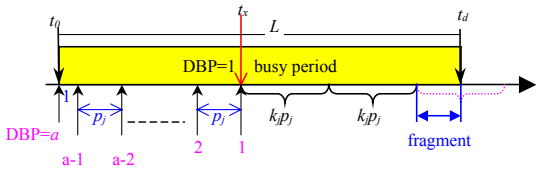

Fig.4: Workload of instances whose DBP>1 at time point t_0.

Assume that, at t_0, the job $a(j)$ has $DBP=a$ ($a>1$). The worst-case is the following situation: after one clock tick, this DBP value will be decreased by one, and then, after every period p_j, the DBP value will be minus one. No instance is executed in the interval $[t_0,t_x]$, where t_x is the time at which $a(j)$ has its DBP=1. We note $l = t_x - t_0$. The worst case correspond to:

$$l = 1 + (DBP_j(t_0) - 2)p_j \qquad (4)$$

For the interval after t_x, the workload evaluation is similar to the one used for the set U. According to (1) and (2), we obtain:

$$W^1_{a()-U} = \sum_{j \in a()-U} \left(\left\lfloor \frac{L-l}{k_j p_j} \right\rfloor m_j \right) c_j \qquad (5)$$

$$W^2_{a()-U} = \sum_{j \in a()-U} Min\left(\left\lceil \frac{(L-l) - k_j p_j \left\lfloor \frac{L-l}{k_j p_j} \right\rfloor}{p_j} \right\rceil, m_j \right) c_j \qquad (6)$$

Formula (7) expresses the total workload of the jobs in the set ($a() - U$).

$$W_{a()-U} = W^1_{a()-U} + W^2_{a()-U} \qquad (7)$$

By using equation (3) and (7), the total workload of a DBP=1 busy period is:

$$W = W_U + W_{a()-U} \qquad (8)$$

Now, we consider the second situation where a job with DBP=1 is blocked by the running instance of a job whose DBP>1(non pre-emption) as shown in Fig. 5. DBP=1 instance generates a workload starting at time point t_y, but a DBP>1 instance of $a(i)$ has occupied the server at t_j and is still being executed. Because of non pre-emption, DBP=1 instances can only be executed after the completion of the DBP>1 instance.

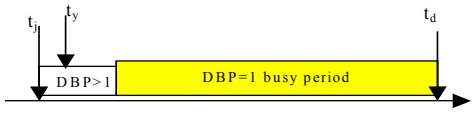

Fig. 5: DBP=1 busy period blocked by DBP>1 instance

Then, we calculate the tight upper bound of the workload in the interval $L = [t_j, t_d]$ (see Fig. 6). Obviously, at the completion of an instance of *a(i)*, the DBP of the next instance is still greater than 1 and it will not be executed in DBP=1 busy period. After p_i (at time t_r), *a(i)* also generates a workload only if DBP=1. So, the worst-case is when *a(i)* has DBP=2 at time t_j. For whatever (m_i,k_i)-firm constraint of *a(i),* the worst-case is at time point t_r, the DBP value changes to 1 as in Fig. 6.

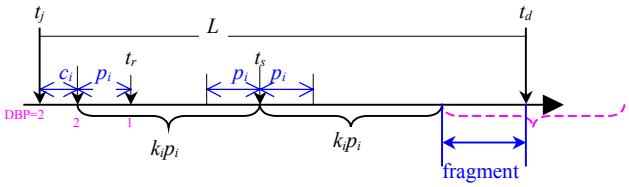

Fig. 6: DBP>1 instance causes workload in DBP=1 busy period

DBP=1 workload of *a(i)* in $L = [t_j, t_d]$ is calculated as follows. Let $t_s = t_j+c_i+k_ip_i$. Then we can predict that the instance invocated within $[t_s, t_s+p_i]$ has DBP>1. Otherwise, if this instance has DBP=1, there will be m_i-*1* DBP=1 instances in $[t_r, t_s]$ (as in every k_ip_i there are, at most, m_i DBP=1 instances in DBP=1 busy period). Moreover, as the instance within $[t_r-p_i, t_r]$ is a DBP>1 instance and it has not been executed, then in $[t_r-p_i, t_s] = k_ip_i$ there are only m_i-*1* DBP=1 instances. However, this violates the (m_i,k_i)-firm. So, we can conclude that the instance invocated within $[t_s,t_s+p_i]$ is a DBP>1 instance and will not be executed. We can extend this to the instances invocated in $\left[t_s+k_ip_i, t_s+(k_i+1)p_i\right]$, etc.

With the same reasoning, we can predict that the instance invocated in $[t_s-p_i, t_s]$ has DBP=1. Otherwise, in $[t_r, t_s-p_i]$, there will be m_i DBP=1 instances. This is in violation of the fact that in $[t_j, t_j+c_i]$, there is already one executed instance, and there will be only m_i-*1* DBP=1 instances in $[t_r-p_i,t_s-p_i]$. This reasoning can be extended to instances in $\left[t_s+k_ip_i, t_s+(k_i-1)p_i\right]$ as well.

For *a(i)*, after the first DBP>1 instance is executed and in $[t_j+c_i, t_d]$, it generates a workload of m_ic_i in every k_ip_i; this is given in equation (9):

$$W^1_{a(i)} = \left(\left\lfloor \frac{L-c_i}{k_ip_i} \right\rfloor^+ m_i+1\right)c_i \qquad (9)$$

where $x^+ = max(0,x)$.

In the fragment (Fig. 6), the *a(i)* workload is given by:

$$W^2_{a(i)} = Min\left(\frac{L-c_i-\left\lfloor \frac{L-c_i}{k_ip_i}\right\rfloor k_ip_i}{p_i}-1, m_i-1\right)^+ c_i \qquad (10)$$

So, the total workload caused by *a(i)* is $W_{a(i)} = W^1_{a(i)} + W^2_{a(i)}$.

Now, we have to take into account all other jobs with DBP=1 at time t_y. The worst-case is for t_y, such as $[t_j, t_y] = 1$ (see Fig 7). In $[t_y, t_d]$, we calculate the same as in equations (1) to (7). Some DBP=1 invocations occur, at the earliest, at time t_y, while other jobs may have DBP=1 invocations after t_y (as shown in Fig 8).

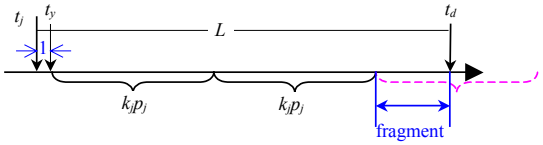

Fig 7: DBP=1 invocations starting time sub-situation-1

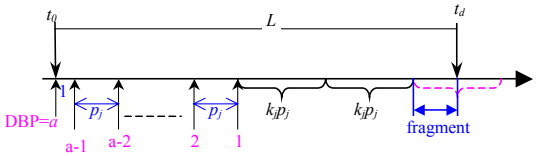

Fig 8: DBP=1 invocations starting time sub-situation-2

Jobs *a(j)* ($j \neq i$) with DBP=1 starting, at the earliest, at t_0+l (with $l=1+(DBP_j(t_j)-2)p_j$). The method used to calculate equations (5) and (6) is applied to obtain the total workload caused by *a(j)*, and gives formula (11):

$$W_{a()-a(i)} = \sum_{a(j)\in a()-a(i)} \left(\left\lfloor \frac{L-l}{k_jp_j} \right\rfloor m_jc_j + Min\left(\left\lfloor \frac{(L-l)-k_jp_j\left\lfloor \frac{L-l}{k_jp_j}\right\rfloor}{p_j} \right\rfloor, m_j \right) c_j \right) \qquad (11)$$

Then, the total workload is :

$$W' = W_{a(i)} + W_{a()-a(i)} \qquad (12)$$

3.3 Theorem

Theorem: Let *a()* be a set of periodic jobs, $a()=\{a(1),a(2),...,a(n)\}$ $(n>1)$, where $a(i)=(c_i, p_i, m_i, k_i)$, $d_i = p_i$ (see section 2.1) If the job set *a()* satisfies the following conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will schedule any concrete set of

periodic jobs generated from *a()*, i.e. there will not be any violation of the (m_i, k_i)-firm constraints.

C_1: for any arbitrary time length L: $W \le L$

C_2: $\forall i, \forall L, L > min_i(p_i)$: $W' \le L$

where W is given by equation 8 and W' by equation 12.

Proof:

The proof is by contradiction. Assume the contrary, i.e., that *a()* satisfies condition C_1 and C_2 from the theorem, and that there is a concrete set of periodic jobs ω_s generated from *a()*, such that a job in ω_s falls into failure state, i.e. ω_s has violated the *(m, k)*-firm guarantee.

We analyze the process of falling into the failure state. Intuitively, the violation of the (m, k)-firm guarantee will happen after a time interval from time 0. Let t_d be the earliest time point where ω_s falls into failure state.

Obviously, only **DBP=1 busy period** leads to (m,k)-firm violation. Starting time t_d we work our way backwards to discover which cases occurred relative to this last DBP=1 busy period, knowing that, for all the possibilities, there are only three cases we could find:

Case 1) DBP=1 busy period starts from an idle time, and all executed jobs have the deadlines before t_d.

Case 2) DBP=1 busy period is blocked by a DBP>1 instance, and all executed jobs have deadlines before t_d.

Case 3) There are some job invocations which have deadlines after t_d.

Case 1: In this case, we go backward from t_d to the starting time of this DBP=1 busy period.

This situation happens because, before this DBP=1 busy period, all of the workload has been completed or some workload remains which could not be finished before its deadline and it can be discarded in its tolerable region. So, there is an idle time between this DBP=1 busy period and the previous DBP busy period. (The critical situation is that a DBP=1 instance starts just at the end of the previous busy period, i.e., this idle time is 0. But this does not influence our analysis.) Let L be the length from the starting time t_0 of DBP=1 busy period to the violation time point t_d.

As there is no other idle time in this DBP=1 busy period $[t_0, t_d]$, the total workload in DBP=1 busy period is presented by (8). Moreover, since the system falls into a failure state at t_d, we can say that the total workload is definitely greater than the time interval L, (i.e., $W>L$). However, this contradicts condition C_1 and it establishes the theorem for Case1.

Case 2: In this case, we go back to the start time of the instance which blocked the last DBP=1 busy period. Let L be the time length from the violation time t_d to the identified time point.

In the interval L, the worst-case of the workload is presented by the formula (12). Since in L there is no idle time (otherwise, our analysis of Case 1 would be directly applied to the interval), and the system falls into a failure state at time point t_d, so $W' > L$. It contradicts our condition C_2 and establishes the theorem for Case 2.

Case 3: In this case, we go back to the last job which had an instance occurring prior to t_d with a deadline after t_d.

Let *a(i)* be this last job and L ($min_{j \neq i}(p_j) < L < p_i$) be this time length, shown in Fig 9.

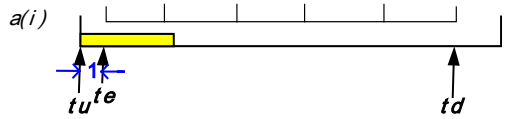

Fig. 9 DBP=1 busy period is blocked by a instance with the deadline after td

The workload of job *a(i)* in $[t_u, t_d]$ is c_i. All the other jobs *a(j)* ($a(j) \in a()-a(i)$) have their deadlines before t_d can add the workload to this period, and their DBP value is superior to 1 at time t_u. Formula (11) gives the workload of *a(j)*.

Note that the determined time length is limited with $min_{j \neq i}(p_j) < L < p_i$, and if we use this special value in formula (9) and (10), we can derive that $W_{a(i)} = c_i$. Assuming that a concrete job set leads to a failure state and that there is no idle time in L, we obtain $W' > L$; it contradicts the condition C_2 as well. This establishes the theorem for Case 3.

End of Proof.

Corollary: Let *a()* be a set of sporadic jobs, $a()=\{a(1),a(2),...,a(n)\}$ ($n>1$), where $a(i)=(c_i, p_i, m_i, k_i)$, $d_i = p_i$. If the job set *a()* satisfies conditions C_1 and C_2 in any time interval, then NP-DBP-EDF will be able to schedule any concrete set of *sporadic jobs* generated from *a()*, i.e. there will be not violation of the (m_i, k_i)-firm constraints.

Proof: As the worst case behavior of a sporadic job ("worst" in the sense of requiring the most processing time) occurs when *a(i)* behaves like a periodic job, that is, *a(i)* has invocated every p_i time step. Remember that a sporadic job can behave as periodic job. Therefore, if conditions C_1 and C_2 are satisfied, NP-DBP-EDF algorithm can schedule any concrete set generated from a periodic job set. As we have defined, the arrival curve and the workload of any sporadic job set are always inferior to the periodic concrete set. Whenever a failure state happens, the two conditions have been violated. So, the conditions are also sufficient to guarantee that NP-DBP-EDF will be able to schedule any concrete set generated from a sporadic job set.

End of proof.

3.4 Sufficient verification length

As has been shown, in our sufficient condition for NP-DBP-EDF scheduling, all $DBP_j(t)$ are a function of

time. Therefore, an interval is necessary to indicate the time evaluation domain. That is to say, we need a ***sufficient length*** for terminating the verification of our sufficient condition.

First, we explain the following definitions:

1) all possible DBP values for one job *a(i)* with (m,k)-firm constraint:

All DBP values appearing in the scheduling sequence are limited to a natural number in [*0, k_i-m_i+1*], but not every value will appear in a concrete situation. Because the system falls into a failure state when DBP=0 instance appears, the successful sequence under consideration (no failure state contained sequence) contains DBP values which are limited in [*1, k_i-m_i+1*]. For a job, *a(i)* has k_i-m_i+1 DBP values in a successful scheduling sequence. In any k_i-m_i+2 instances of *a(i)*, there must be at least two invocations which have the same DBP value in a successful sequence.

2) for a job set with *n* jobs, *at one time point*, it has $\prod_{i=1}^{n}(k_i-m_i+1)$ successful DBP configuration possibilities.

3) for a strict periodic job set, the inter-distribution of the instances reappears after each LCM (Least Common Multiple) of $\{p_1,\ldots,p_n\}$. Suppose that the time points $t_1,t_2,\ldots t_x$ ($x \in N$) are the time points with interval LCM, i.e., $t_{i+1} = t_i + LCM\ i \in (1\ldots x)$. Not considering the concrete possibilities, at all time points of $t_1,t_2,\ldots,t_x$, there are at most $\prod_{i=1}^{n}(k_i-m_i+1)$ possible successful DBP configurations. So, in $x=\prod_{i=1}^{n}(k_i-m_i+1)+1$, there must be at least two time points where all instances of the jobs have the same DBP values. And our scheduling can repeat on with the same successive scheduling from the two time points, because at each time point $t_1,t_2,\ldots,t_x$ the inter-distribution of the instances is always the same.

Finally, we can conclude that the sufficient length L_{max} for terminating the verification of our sufficient condition (only for strict periodic job set) is:

$$L_{max} = r + \left(\prod_{i=1}^{n}(k_i-m_i+1)+1\right) \mathrm{LCM} \qquad (13)$$

where *r* is the last release time.

Obviously, this is a sufficient but not necessary length, because we are considering it from an aspect of permutation. Once at a time point the DBP values of all instances are the reappearance of DBP values, which occurred at a certain LCMs before, the schedulability can already be given. Since in this case, the following sequence will be the iteration of the sequence which took place between the two time points. In practice the test can stop earlier as soon as the repetition occurs for the first time at a multiple of LCM time point.

4. Applications of the sufficient condition

In this section, we apply our sufficient condition to dimensioning the sufficient server capacity for guaranteeing the (m,k)-firm constraint in contrast to that of (k,k)-firm (i.e. HRT). In practice, the dimensioning can be done not only off-line but also on-line. For example, a network supporting real-time QoS should be based on the sufficient condition to decide the acceptance or rejection of a new job (or stream) in its connection admission control procedure; an adaptive real-time system could go from a nominal mode corresponding to (k,k)-firm to a degraded mode, still ensuring (m,k)-firm constraint with the presence of some resource failures.

4.1 Bandwidth dimensioning for graceful degradation of QoS

Consider the following networked control system (Fig. 10) where four sensors are connected to a controller via an intranet.

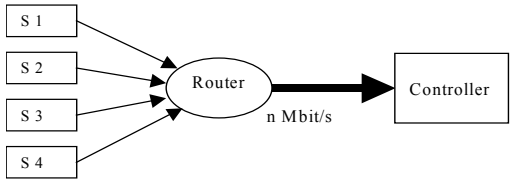

Fig. 10: Application Model

Assuming Intserv/RSVP is used at the entrance router[1], the router should reserve a certain bandwidth for guaranteeing real-time QoS. The data packets of sensors should be transmitted to the controller or discarded (during peak network traffic load period but within the (m,k)-firm constraint tolerance region) before the next packet arrives from the same sensor (i.e., deadline is equal to period). The configuration in terms of inter-arrival time, packet size and specified (m,k)-firm constraints is given in Table 1.

	Packet size (kbit)	Interarrival time (ms)	(m,k)-firm constraint
S1	8	12	(2,5)
S2	8	20	(4,5)
S3	1	5	(1,4)
S4	4	6	(1,5)

Table 1: Parameters of the sources

[1] Despite the scalability problem, we still believe that IntServ QoS architecture can be used within an IP network for factory communication needs since, with its limited size, the scalability problem is not critical.

We will demonstrate the difference in terms of the minimum bandwidth that the router must reserve to deterministically guarantee (k,k)-firm and (m,k)-firm constraints. We start our scenario with the worst case DBP values (as mentioned in cases 1 to 3 in section 3.3).

We calculated the cumulative workload during a sufficient schedulability analysis length of (m_i,k_i)-firm (i = *1, 2, 3, 4*). Fig. 11 shows how the router load changes according to the time length L (here for the concrete task set, L_{max}=9600ms). The upper curve corresponds to the maximum workload of (k,k)-firm and the lower one corresponds to that of (m,k)-firm. This figure shows that on the average, as shown in the slopes of the lines, (k,k)-firm requires a bandwidth of 1.93 Mbit/s, while (m,k)-firm only requires 0.8 Mbit/s.

Now, to determine the sufficient bandwidth we need to find the greatest upper bound slope. This is given by the highest value of the workload divided by time length L. For both (k,k)-firm and (m,k)-firm cases in our example, these values are found at the beginning of our calculation. Fig. 12 shows the cumulative workload in units of kbit during the first 20ms (a detailed initial view of Fig. 11).

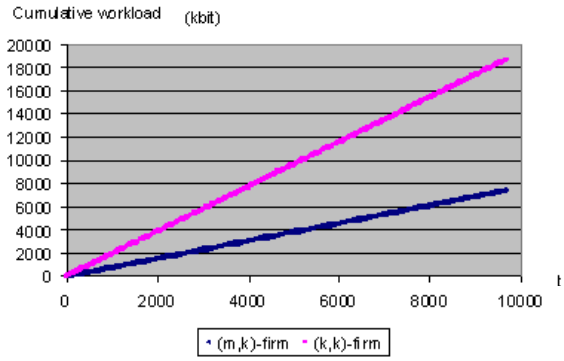

Fig. 11: Router load in average sense

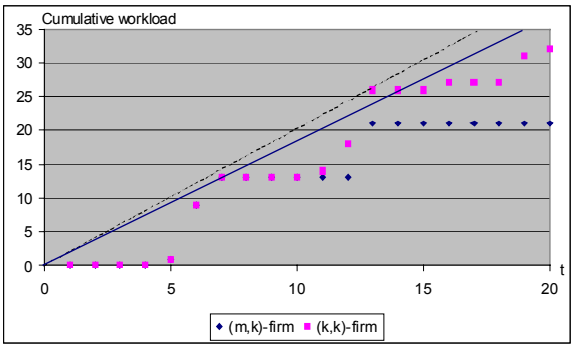

Fig. 12: Deterministically guarantee dimensioning

To guarantee (k,k)-firm, or HRT constraint, Jeffay's conditions must be satisfied. The minimum bandwidth for HRT guarantee is presented by the dotted line upper bounding the (k,k)-firm workload curve (Fig. 12). So, the sufficient bandwidth dimensioned by Jeffay's condition is 2Mbit/s.

For our sufficient conditions of NP-DBP-EDF, there are some concrete parameters such as the set U and DBP(t). We do not take into account the actual distribution of the jobs, which can be either inside or outside the set U, but include all DBP values starting with the worst values, without considering their roles in a concrete situation (at a critical time point, DBP=2 and changes to DBP=1 after only one time-click). The sufficient bandwidth for (m,k)-firm constraint is presented by the continuous line (Fig. 12). The sufficient bandwidth dimensioned by our sufficient condition is 1.857Mbit/s instead of 2Mbit/s. As calculated, even with the arbitrarily selected parameters, our sufficient condition can still economise 7.15% of the bandwidth.

Using WCIP [8], we can easily understand that the worst execution interference with (m,k)-firm constraint could support the most mandatory consecutive m invocations among the consecutive k invocations of another job. That is why the simulation result of our sufficient condition was not dramatically decreased from that of Jeffay's condition in terms of the sufficient bandwidth. But, if we calculate in a concrete situation until we get the sufficient length shown in formula (13), we will have a much smaller sufficient bandwidth. However, for the absolute guarantee in pre-dimensioning, we must take into account the worst-case router load, which takes place at the initial time region, as shown in Fig 12.

4.2 Overload management in automotive control applications

In this part we show how our sufficient condition can help the dimensioning of the processor capacity in an automotive control system for making it fault-tolerant while using reduced resources.

In in-vehicle embedded system design, the current trend is to use generic processors to replace the specific ones [20]. To achieve this goal, OSEK is defined by carmakers and the ECU (Electronic Control Unit) suppliers as the standard operating system [17]. Moreover, the effort to establish a common platform for supporting portable software modules is continuing inside the AUTOSAR consortium [http://www.autosar.org/]. One of the objectives is to be able to run a car function (e.g. engine control, ABS, etc.) over any generic processor, thus ensuring fault-tolerance when the same function is replicated on more than one processor. All the ECUs are interconnected via a bus (e.g. CAN [18] or FlexRay [19] in the near future).

For making the system fault-tolerant, the classical approach consists in reserving the sufficient spare

capacity so that the tasks can be reassigned or re-executed on fault-free processors upon failure detection; without violating any deadlines (i.e. (k,k)-firm). As indicated in our introduction, the drawback of this approach is that the system resources are often underutilized when no faults are present. For the automotive industry where the cost constraints are omnipresent, this approach has not always been acceptable. The approach based on the (m,k)-firm model is more suitable. It consists in invoking an overload management technique upon detection of a failure [9]. Following this approach the system can still work with the presence of some processor failures without necessarily reserving as many resources as used in the classic approach.

The simulation is implemented by taking a case study similar to that of Ramanathan [9], in which the author has shown that the control laws of the automotive control applications can tolerate some deadline misses specified by the (m,k) patterns, without leading to a dangerous situation for the vehicle. Based on our experience in automotive systems [20], [6], we add another argument that most control loops are based on over-sampling input data (sensor data) to increase dependability. The occasional loss of some input data will not automatically lead to a dangerous situation.

We then consider a system (Fig.13) composed of four control functions: cruise control, traction control, braking control and engine control. At first, all four functions are implemented on the four ECU of the system, but only one function is running on each ECU. In case of failure of an ECU, the corresponding function it ensures is woken up on one of the remaining ECU, thus tolerating an ECU failure.

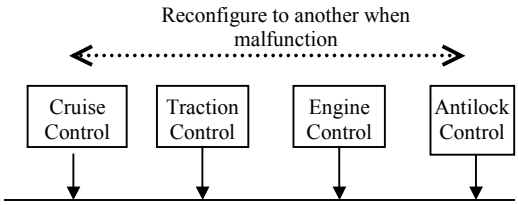

Fig. 13: Vehicle control system model

In what follows, we just consider the extreme case of three simultaneous ECU failures. Our goal is to dimension the processor capacity of an ECU to continue to guarantee meeting of the (m,k)-firm constraint of the four functions. The deadline miss tolerated by each function is assumed to be as given in Table 2.

	Execution time (ms)	Task period (ms)	(m,k)-firm constraint
Antilock control	2	20	(1,4)
Traction control	6	30	(1,4)
Engine control	5	50	(1,4)
Cruise control	6	100	(2,3)

Table 2: task parameters of control system in vehicle

The target (m,k)-firm constraint for each function can be obtained either by following the control law stability/tolerance study method of [9] or by measuring and simulating the car situations in presence of failures (fault injection) [6].

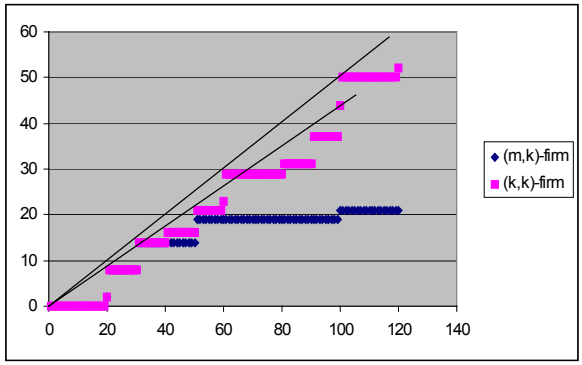

Fig. 14: Workload of (k,k)-firm in contrast of (m,k)-firm for dimensioning system sufficient capacity

The upper line with the slope value 0.495 is the sufficient capacity for the HRT measured by Jeffay's condition. The lower one with the slope value 0.42 is the sufficient capacity for the fault tolerant system in the form of (m,k)-firm. This represents a saving ratio of 15%.

4.3 Discussion on the limits of the deterministic (m,k)-firm guarantee

As one can see from the above examples, the advantage of using (m,k)-firm, compared with (k,k)-firm, is not always noticeable. In fact, our sufficient condition and that of Jeffay can even be overlapped in some situations, thus forcing the service of all *k* jobs even though the system is only under (m,k)-firm constraint. To understand that, let us first take the following numerical example given in Table 3. This configuration is derived from that of Table 1 with some modifications. Four streams (jobs) with (m_i,k_i)-firm

constraints should be executed by the MIQSS model server.

	(m,k)-firm constraint	Processing Time	Period/ Deadline
Stream 1	(2,5)	8	12
Stream 2	(4,5)	10	20
Stream 3	(3,6)	2	5
Stream 4	(1,5)	4	6

Table 3: Parameters of Periodic Job Set

Fig. 15 and Fig. 16 give the cumulative processor demand in time for, respectively conditions C_1 and C_2 in the case of HRT and (m,k)-firm of the concrete job set in Table 2. The x-coordinate represents the time interval (L), and the y-coordinate represents the processor demand which must be executed before the end of L. So, we calculate the changes of this processor demand according to the length of the time interval. In Fig. 15 and 16 the upper trapezium (solid line) represents the result of HRT under NP-EDF, and the lower one (dashed line) that with (m,k)-firm constraint under NP-DBP-EDF. To start simulation, we assum the worst case for (m,k)-firm by setting all $DBP_i(t) = 1$.

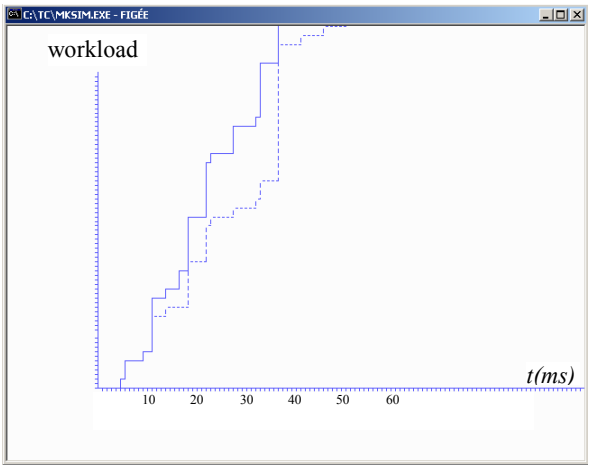

Fig.15: Difference between conditions 1

From Fig. 15 and 16, we can see that there is an overlap at the initial time.

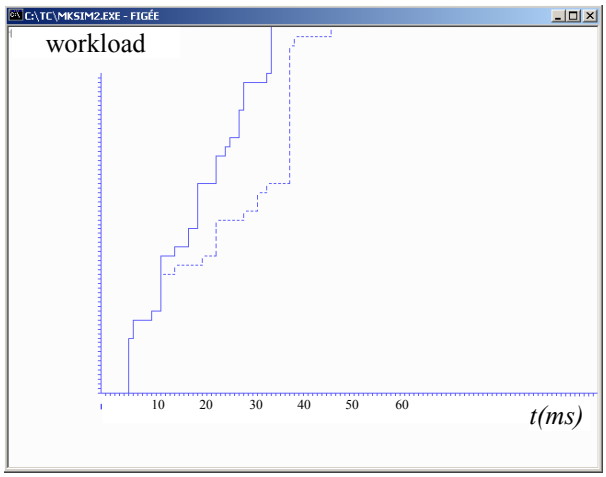

Fig.16: Difference between conditions 2

In fact, condition C_1 of our theorem can be transformed to be like the condition (1) in Jeffay's theorem. Assuming that all jobs of all sources are within the set U (the worst case), and the interval L is less than $\min(m_i \cdot p_i)$, we get $\left\lfloor \frac{L}{k_i p_i} \right\rfloor = 0$ and the term $Min\left(\left\lfloor \frac{L - k_i p_i \left\lfloor \frac{L}{k_i p_i} \right\rfloor}{p_i} \right\rfloor, m_i \right) c_i = \left\lfloor \frac{L}{p_i} \right\rfloor c_i$. The condition C_1 of our theorem has been transformed to the condition (1) in Jeffay's theorem.

Condition C_2 of our theorem can also be transformed to be like condition (2) in Jeffay's theorem.

As we have interpreted, the sufficient condition of (m,k)-firm is always under the bound of Jeffay's theorem, and can reach the bound of Jeffay's theorem with some assumptions and forced evaluation conditions. Notice that these assumptions and forced evaluation conditions can be realized, or not in concrete situations. However, this limits the advantage of using the (m,k)-firm tolerance compared with a system only requiring statistic (m,k)-firm guarantee.

We have proven in our report [1] that DBP scheduling may fail into failure state even with arbitrary low utilisation. The same problem has also revealed for DWCS algorithm [15]. As in HRT, (m,k)-firm schedulability remains still NP-hard. That is why our simulation result is pessimistic. But if we do it on-line within the time length of formula (13), a significant efficiency can be obtained. Furthermore, the recent proposal of [14], which relaxes the (m,k)-firm constraint by defining the virtual deadline concept, consists in an interesting way to improve the advantage of the (m,k)-firm system in terms of relaxing the resource need compared with the system requirements under (k,k)-firm.

5. Conclusions

In this paper, we first explained how (m,k)-firm model can be used to define the graceful degradation of real-time QoS, thus allowing the fault-tolerance, and then addressed the problem of the deterministic guarantee of (m,k)-firm real-time requirements for a set of periodic or sporadic jobs sharing a common server. DBP has been chosen for its interesting feature of dynamically assigning priorities based on the previous history of the system (k-sequence). This makes it suitable for QoS management in adaptive real-time systems and networks. Our main contribution is having given the expression of the sufficient condition under NP-DBP-EDF scheduling for deterministically guaranteeing (m,k)-firm constraint. This result is necessary for system server capacity dimensioning. Our future work aims at two complementary directions: 1) research of conditions to avoid the overlapping of the sufficient condition of (m,k)-firm with that of (k,k)-firm and the new job models allowing the improvement with advantages gained with (m,k)-firm in terms of relaxing resource requirements 2) the implementation of dynamic algorithms such as DBP, in terms of admission control procedures, within IP networks (e.g. Internet-based control systems, remote control and monitoring systems based on Internet and power line networks such as what has been proposed in the REMPLI project www.rempli.org) for dynamically managing real-time QoS according to (m,k)-firm model.

References

[1] J. LI. *Sufficient Condition for Guaranteeing (m,k)-firm Real-Time Requirement Under NP-DBP-EDF Scheduling*. Technical report No. A03-R-452, Stage de DEA, LORIA, Jun, 2003.

[2] G. Bernat, A. Burns and A. Llamosi, "Weakly-hard real-time systems", IEEE Transactions on Computers, 50(4), pp.308-321, April 2001.

[3] G. Bernat, A. Burns and A. Llamosi, "Weakly-hard real-time systems", *IEEE Transactions on Computer*s, 50(4), pp.308-321, April 2001.[4] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-firm deadlines", IEEE Transactions on Computers, 44(4), pp1443–1451, Dec.1995.

[5] K. Jeffay, D.F. Stanat, and C.U. Martel, "On Non-Preemptive Scheduling of Periodic and Sporadic Task", IEEE real-time systems symposium, pp129-139, San Antonio (USA), Dec. 4-6, 1991.

[6] C. Wilwert, Y.Q. Song, F. Simonot-Lion, T. Clément, "Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems". *In 9th IEEE International Conference on Emerging Technologies and Factory Automation -EFTA'2003.* (Lisbonne, Portugal). IEEE, 2003. vol 1. pp.193-200.

[7] E.-M. Poggi, Y.-Q. Song, A. Koubaa, Z. Wang, "Matrix-DBP For (m, k)-firm Real-Time Guarantee", RTS2003, pp457-482, Paris (France), 1-3 April 2003.

[8] G. Quan and X. Hu, "Enhanced Fixed-priority Scheduling with (m, k)-firm Guarantee", Proc. Of 21st IEEE Real-Time Systems Symposium, pp.79-88, Orlando, Florida, (USA), November 27-30, 2000.

[9] P. Ramanathan, "Overload Management in Real-Time Control Application Using (m, k)-Firm Guarantee", IEEE Transactions on Parallel and Distributed Systems, 10(6):549–559, June 1999.

[10] Z. Wang, Y.Q. Song, E.M. POGGI and Y.X. Sun, "Survey of Weakly-Hard Real-Time Scheduling Theory and Its Application", International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Wuxi (china), Dec. 16-20, 2002.

[11] F. Wang and P. Mohapatra, "Using differentiated services to support Internet telephony", Computer communications, Vol.24, Issue18, pp1846-1854, Dec. 2001.

[12] R. West and C. Poellabauer, "Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams", Proc. of 21st IEEE Real-Time Systems Symposium, pp239-248, Orlando, Florida, (USA), November 27-30, 2000.

[13] Richard West, Yuting Zhang, Karsten Schwan and Christian Poellabauer, "Dynamic Window-Constrained Scheduling of Real-Time Streams in Media Servers", *IEEE Transactions on Computers*, Volume 53, Number 6, pp. 744-759, June 2004

[14] Yuting Zhang, Richard West and Xin Qi, "A Virtual Deadline Scheduler for Window-Constrained Service Guarantees", in *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS)*, December 2004.

[15] Mok, A.K. and W. R. Wang, "Window-Constrained Real-Time Periodic Task Scheduling", *22nd IEEE Real-Time Systems Symposium (RTSS'01)*, pp15-24, London, England, December 03 - 06, 2001.

[16] G.Bernat "Response Time Analysis of Asynchronous Real-Time Systems"; Real-Time Systems, 25, 131-156, 2003

[17] OSEK, *OSEK/VDX operating system, version 2.2*, 2001. http://www.osek-vdx.org

[18] ISO, *Road vehicles – Interchange of digital information – Controller area network for high-speed communication*, ISO 11898, International Organization for Standardization (ISO), 1994.

[19] FlexRay Consortium, http://www.flexray.com, 2004.

[20] C. Wilwert, N. Navet, Y.Q. Song, F. Simonot-Lion, "Design of Automotive X-by-Wire Systems", to appear in *The Industrial Communication Technology Handbook* (edited by R. Zurawski), CRC Press, 2005.